Progress-Bar Documentation

The provides a comprehensive progress indicator solution with both circular and linear variants. It supports multiple modes including determinate, indeterminate, buffer, and query states, with smooth animations, customizable colors, and full accessibility support.

How to use

Import the component and configure it with your desired properties.

Circular Progress

Circular progress indicator with customizable size, color, and animation.

```
<aava-progressbar
  [percentage]="25"
  label="25% Complete"
  type="circular"
  [svgSize]="100"
>
</aava-progressbar>

<aava-progressbar
  [percentage]="50"
  label="50% Complete"
  type="circular"
  [svgSize]="100"
>
</aava-progressbar>

<aava-progressbar
  [percentage]="75"
  label="75% Complete"
  type="circular"
  [svgSize]="100"
>
</aava-progressbar>

<aava-progressbar
  [percentage]="100"
  label="100% Complete"
  type="circular"
  [svgSize]="100"
>
</aava-progressbar>
```

Linear Progress

Linear progress bar with support for determinate, indeterminate, and buffer modes.

```
<aava-progressbar [percentage]="25" type="linear" mode="determinate">
</aava-progressbar>

<aava-progressbar [percentage]="50" type="linear" mode="determinate">
</aava-progressbar>

<aava-progressbar [percentage]="75" type="linear" mode="determinate">
</aava-progressbar>

<aava-progressbar [percentage]="100" type="linear" mode="determinate">
</aava-progressbar>
```

Features

Multiple Progress Types

- Circular Progress : SVG-based circular indicator with smooth animations
- Linear Progress : Horizontal progress bar with customizable height and styling
- Responsive Design : Automatically adjusts size based on screen dimensions

Progress Modes

- Determinate : Shows exact progress percentage with smooth animations
- Indeterminate : Animated loading indicator for unknown progress
- Buffer : Shows both progress and buffer values (linear only)
- Query : Animated indicator for querying operations

Customization Options

- Custom colors and themes
- Configurable sizes and positions
- Smooth animations with easing
- Accessibility features with ARIA support

Performance Optimized

- OnPush change detection strategy
- Efficient SVG animations
- Optimized rendering and memory management
- Responsive resize handling

API Reference

Inputs

Property	Type	Default	Description
percentage	number	0	Progress percentage (0-100)
bufferValue	number	0	Buffer value for buffer mode (0-100)

Property	Type	Default	Description
label	string	''	Label text displayed with the progress bar
type	'circular' | 'linear'	'circular'	Type of progress indicator
color	string	'#2E308E'	Color of the progress indicator
mode	'determinate' | 'indeterminate' | 'buffer' | 'query'	'determinate'	Progress mode
svgSize	number	''	Custom SVG size (overrides responsive sizing)
position	'12' | '3' | '6' | '9' | number	'12'	Starting position for circular progress (clock positions)

Outputs

Property	Type	Description
None	-	This component doesn't emit events

Properties

Property	Type	Description
progressId	string	Unique identifier for the progress element
circumference	number	Circumference of the circular progress (readonly)
dashOffset	number	Current dash offset for circular progress
errorMessage	string	Error message for invalid inputs
displayPercentage	number	Animated display percentage for linear progress
rotationAngle	number	Rotation angle for circular progress position

Methods

Method	Parameters	Description
updateProgress()	None	Updates the progress display and animations
writeValue()	value: number	Sets the progress value (ControlValueAccessor)
registerOnChange()	fn: (value: number) => void	Registers change callback (ControlValueAccessor)
registerOnTouched()	fn: () => void	Registers touched callback (ControlValueAccessor)

CSS Custom Properties

The component uses CSS custom properties for dynamic styling:

Circular Progress Properties

Property	Description
--progress-text-weight	Font weight for progress text
--progress-text-color	Text color for progress display
--progress-text-font	Font size for progress text
--progress-label-font	Font size for progress label
--progress-label-color	Color for progress label
--progress-label-line-height	Line height for progress label
--progress-label-weight	Font weight for progress label
--progress-transition	Transition timing for progress animations
--progress-indeterminate-animation	Animation for indeterminate mode

Linear Progress Properties

Property	Description
--progress-linear-height	Height of the linear progress bar
--progress-linear-border-radius	Border radius for linear progress

CSS Classes

The component uses CSS classes for styling and state management:

Container Classes

Class	Description
.progress-container	Main container for circular progress
.linear-progress-container	Main container for linear progress

Progress Classes

Class	Description
.progress-background	Background circle for circular progress
.progress-bar	Main progress circle/bar
.progress-text	Text display inside circular progress
.progress-label	Label text for progress
.linear-bar	Container for linear progress bar
.linear-progress	Main linear progress element
.buffer-bar	Buffer progress element
.indeterminate-bar	Indeterminate progress element
.progress-percentage	Percentage display for linear progress

State Classes

Class	Description
.indeterminate	Indeterminate animation state
.progress-error	Error message styling

Best Practices

Progress Mode Selection

• Use determinate mode when you know the exact progress
• Use indeterminate mode for unknown loading times
• Use buffer mode for operations with both progress and buffer (e.g., video loading)
• Use query mode for search or query operations

Color and Styling

• Choose colors that provide good contrast with the background
• Use consistent colors across your application
• Consider using semantic colors (success, warning, error) for different states
• Ensure accessibility with proper color contrast ratios

Performance

• Avoid frequent percentage updates for smooth animations
• Use appropriate animation durations based on the operation
• Consider using indeterminate mode for very short operations

Accessibility

• Always provide meaningful labels
• The component includes proper ARIA attributes
• Ensure sufficient color contrast
• Test with screen readers

Responsive Design

• The component automatically adjusts size on different screens
• Test on various device sizes
• Consider custom sizing for specific use cases

Accessibility

ARIA Support

• Proper role="progressbar" attribute
• aria-valuenow for current progress value
• aria-valuemin="0" and aria-valuemax="100" for value range
• Screen reader announcements for progress updates

Keyboard Navigation

• Focus indicators for interactive elements
• Proper tab order
• Keyboard-accessible progress controls

Visual Accessibility

• High contrast color options
• Clear visual indicators
• Scalable text and graphics
• Support for reduced motion preferences

Browser Support

• Modern Browsers : Full support for all features
• SVG Support : Required for circular progress
• CSS Animations : Required for smooth transitions
• ES6+ Features : Required for component functionality
• Change Detection : OnPush strategy support